TÉCNICAS DE PROGRAMACIÓN

VARIABLES Y TIPOS

- # ¿QUE ES PYTHON?
- ## EVOLUCIÓN HISTÓRICA:
- ## MANERA DE EJECUTARSE
- ## PARADIGAMA DE PROGRAMACIÓN:
- ## NIVEL DE ABSTRACCIÓN:
- # PROPOSITO:

SIEMPRE ESCRIBIREMOS CODIGO EN INGLES

https://docs.python.org/3.6/library/stdtypes.html

VARIABLES

VARIABLES

```
a = b = c = d = f = "Hello World!"
a
#'Hello World!'
c
#'Hello Worl#d!'
f
#'Hello Worl#d!'
```

EXPLÍCITO

```
int addone(int x) {
    int result = x + 1;
    return result;
}
```

INFERIDO

```
def addone(x):
    result = x + 1
    return result
```

VARIABLES

EL NOMBRADO DE VARIABLES DESCRIPTIVAS Y CONSISTENTES HACE QUE EL SOFTWARE SEA MÁS FÁCIL DE LEER Y ENTENDER.

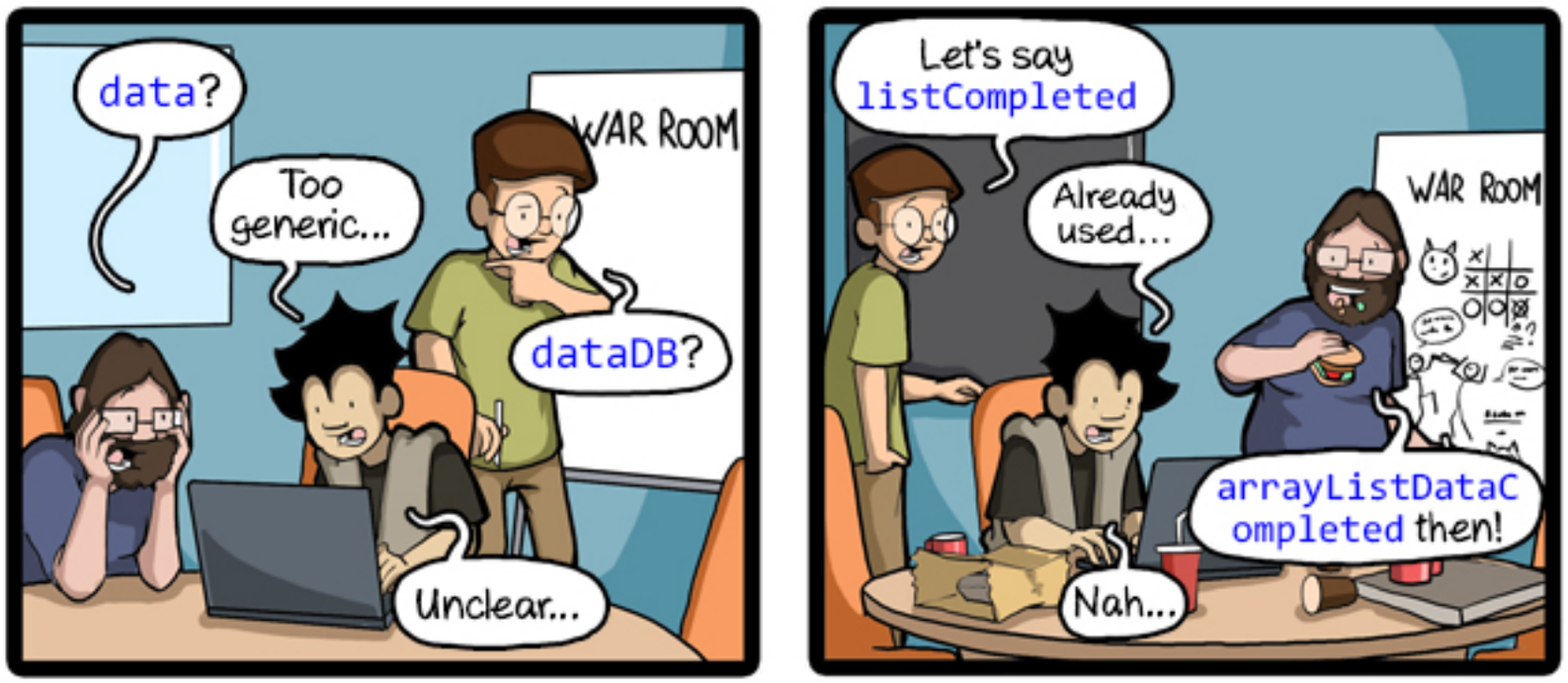
- SIEMPRE ESCRIBIREMOS CÓDIGO EN INGLES
- PRIORIZAR CLARIDAD SOBRE LA BREVEDAD
- EN PYTHON SIMPLE USAR SNAKECASE*
- VARIABLES Y FUNCIONES SIEMPRE INICIAN CON MINÚSCULA
- INCLUIR TODAS LAS PALABRAS NECESARIA PARA ENTENDER
- USAR NOMBRES BASADOS EN ROLES, NO TIPOS
- LOS TIPOS BOOLEANOS DEBEN LEERSE COMO AFIRMACIONES
- EVITAR SOBRECARGAS EN TIPO DE RETORNO
- ELEGIR BUENOS NOMBRES DE PARÁMETROS QUE SIRVAN DE DOCUMENTACIÓN

NOMBRES BUENOS Y MALOS DE VARIABLES

```
#Malos:
    l
    data_structure
    my_list
    info_map
    dictionary_for_the_purpose_of_storing_data_representing_word_definitions

#Bueno:
    user_profile
    menu_options
    word_definitions
```

VARIABLES



CommitStrip.com

EJERCICIO

¿CUALES DE ESTAS ASIGNACIONES SON VALIDAS?

```
2var = "bar"
var-bar = "bar"
foo.loo = "bar"
@var = "bar"
$foo = "bar"
lambda = "bar"
finally = "yes"
pass = "you shall not"
___buga_wuga_ = "@@@huz"
foo_bar = "bar"
```

PRIMITIVE TYPES (TIPOS PRIMITIVOS)

Tipo	Ejemplo	Uso
bool	`life_is_good = True` `hamsters_are_evil = False`	valores falso/verdadero
int, long	`size_of_shoes = 42` `earth_population =`	dígitos sin punto decimal
float	`pi = 3.14159265359`	dígitos com punto decimal
str	`chinese_hi = “algun texto”`	texto
None	`my_new_book = None`	variable bacia sin valor significativo

OPERADORES BÁSICOS

$$a = 3$$

Tipo	Corto	Largo	Resultado
+	`a += 1`	`a = a + 1`	4
-	`a -= 1`	`a = a - 1`	2
*	`a *= 2`	`a = a * 2`	6
/	`a /= 2`	`a = a / 2`	1.5
Modulo	`a %= 2`	`a = a % 2`	1
Exponente	`a **= 2`	`a = a ** 2`	9
Floor division	`a //= 2`	`a = a // 2`	1
Negación	`a = -a`	`a = 0 - a`	-3

EJERCICIO

¿CUAL ES EL RESULTADO DE CADA LINEA?

```
12 % 8 + 12 // 8 * 8          12
12 % 8 + 12 // (8 * 8)        4
(12 % 8) + 12 // 8 * 8        12
12 % (8 + 12) // 8 * 8        8
```

OPERADORES PARA STRING

https://docs.python.org/3.6/library/string.html

```
# Declaracion de String
greeting = "Hola como estas"
greeting2 = 'Hola como "estas" otra vez'
greeting_multiline = """
Hola
como
estas
"""

# longitud de string
print(len(greeting)) # 15

# existe en
print("Hola" in greeting) # True
print("hola" in greeting) # False

# caracteres de string
print(greeting[0]) # H
print(greeting[-1]) # s

# rangos o subStrings
print(greeting[5:9]) # como
print(greeting[:4]) # Hola
print(greeting[5:]) # como estas
```

```
# Operadores
print("hola " + "hola") # "hola hola"
print("hola " * 5) # "hola hola hola hola hola "

# String Interpolations
print("Hola {0}, como estas".format("Miguel"))
print("Hola {0}, {1}".format("Miguel", "¿Como esta tu dia?"))
print("Hola {}, {}".format("Miguel", "¿Como esta tu dia?"))
print("""
Miguel es: {0},
Diana es: {1},
Juan es: {0}
""".format("Hombre","Mujer"))

for i in range(0,9):
    print("Index: {0:2}, two: {1:4}, three: {2:<4} ,".format(i, i*i, i**3))

print("Miguel, tiene %d, %s" % (24, "hola"))

print("Imaginario %12f" % (22/7))
print("Imaginario %.2f" % (22/7))
print("Imaginario %12.5f" % (222/7))

for i in range(0,9):
    print("Index: %d, two: %4d, three: %4d ," % (i, i*i, i**3))
```

operadores_de_strings

COMPARADORES Y OPERADORES BOLEAMOS

Tipo	Ejemplo	Resultado
<	`1 < 2, 4 < 3`	`True, False`
<=	`2 <= 2, 4 <= 3`	`True, False`
>	`2 > 1, 3 > 3`	`True, False`
>=	`2 >= 2, 3 >= 4`	`True, False`
==	`"foo" == “foo”, 0 == 1`	`True, False`
!=	`"foo" != “bar”, 11 != (10 + 1)`	`True, False`
and	`3 and “foo”, None and True`	`True, False`
or	`1 or None, None or False`	`True, False`
not	`not None, not True`	`True, False`

EJERCICIO

¿CUAL ES EL RESULTADO DE CADA LINEA?

```
2 * 6 + 1 < 2 * (6 + 1)                                  # True
2 * 6 + 1 == 2 * (6 + 1) - 1                             # True
2 ** 3 == 8 and 2 ** 2 == 16                             # False
(2 ** 2 == 4 and False) or (not (16 < 15 or 1 == 3))     # True
```

FUNCIONES

```
def greeting(name):
    return 'Hola {}'.format(name)

print(greeting('Miguel'))
```

funciones

CONVENCIÓN DE NOMBRAMIENTO DE FUNCIONES

- NO USAR NOMBRES DEMASIADO GENERALES O DEMASIADO LARGOS.
- POR CONVENCIÓN DECLARAR EL NOMBRE EN SNAKECASE
- SIEMPRE SE ESCRIBEN CON MINÚSCULAS
- INCLUYENDO TODAS LAS PALABRAS NECESARIAS, MIENTRAS QUE OMITE PALABRAS INNECESARIAS
- LUCHANDO POR UN USO FLUIDO
- COMENZANDO LOS MÉTODOS DE FÁBRICA CON MAKE
- ELEGIR BUENOS NOMBRES DE PARÁMETROS QUE SIRVAN DE DOCUMENTACIÓN
- MÉTODOS DE NOMENCLATURA PARA SUS EFECTOS SECUNDARIOS
- LOS MÉTODOS VERBALES SIGUEN LA REGLA -ED, -ING PARA LA VERSIÓN NO MUTANTE
- LOS MÉTODOS NOMINALES SIGUEN LA REGLA FORMX PARA LA VERSIÓN MUTANTE

COMENTARIOS

```
# Comemtario en linea
```

```
//When I wrote this, only God and I understood what I was doing
//Now, God only knows
```

```
//
// Dear maintainer:
//
// Once you are done trying to 'optimize' this routine,
// and have realized what a terrible mistake that was,
// please increment the following counter as a warning
// to the next guy:
//
// total_hours_wasted_here = 39
//
```

COMENTARIOS

```
facade.registerProxy(new ChannelProxy());
facade.registerProxy(new SoundProxy());
//facade.registerProxy(new SoundAssetProxy());
facade.registerProxy(new SubtitleStateProxy());

facade.registerProxy(new SocialShareProxy());

facade.registerProxy(new TrackProxy());
```

NO APORTAN UN SIGNIFICADO O VALOR AL CÓDIGO

COMPLICAN LA LECTURA

```
public class AbstractModuleMediator extends Mediator implements IMediator {
    /* public variables and consts */

    /* protected and private variables and consts */

    private static const log : Logger = LogContext.getLogger(AbstractModuleMediator);

    /* public functions */

    /**
     * Constructor for the AbstractModuleMediator class
     */
    public function AbstractModuleMediator(name : String, viewComponent : IModule) {
        super(name, viewComponent);
    }
}
```

```
/**
 * Always returns true.
 */
public boolean isAvailable() {
    return false;
}
```

NO SE ACTUALIZAN CONTANTEMENTE

EXTRA

```
// somedev1 -  6/7/02 Adding temporary tracking of Login screen
// somedev2 -  5/22/07 Temporary my ass
```

CONVENCIÓN DE COMENTARIOS

- SIEMPRE TRATAR DE EXPRESARSE EN CÓDIGO SIN COMENTARIOS
- NO SER REDUNDANTE O SER OBVIO
- SIEMPRE USAR COMENTARIOS EN LINEA, Y NO EN BLOQUE
- SI VAS A ELIMINAR CODIGO, NO LO COMENTES SOLO ELIMINADO
- USALOS PARA DESCRIBIR LA INTENCIÓN O LA RAZÓN POR LA CUAL EXISTE UN PEDAZO DE CÓDIGO.
- USALOS COMO SEÑAL DE ALERTA PARA OTROS USUARIOS
- MANTENLOS ACTUALIZADOS

1. ¿QUÉ ES PYTHON?
2. TIPADO INFERIDO Y EXPLÍCITO
3. COMO CREAR VARIABLES
4. BUENAS PRACTICAS DE NOMBRADO DE VARIABLES
5. OPERADORES NUMÉRICOS
6. OPERADORES DE STRINGS
7. OPERADORES LÓGICOS
8. DECLARACIÓN DE FUNCIONES
9. BUENAS PRACTICAS DE NOMBRADO DE FUNCIONES
10. COMENTARIOS
11. BUENAS PRACTICAS DE USO DE COMENTARIOS